The Hermes Framework User's Manual

Éamonn Linehan, Mike Spence

hermes.dsg.cs.tcd.ie

The Hermes Framework User's Manual

Version 1.0 (July 2007)

Copyright © Distributed Systems Group, Department of Computer Science, Trinity College Dublin 2007. All rights reserved.

Comments may be addressed to: Dr, Siobhán Clarke
Distributed Systems Group,
Department of Computer Science,
Trinity College,
Dublin 2.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license available for download from http://hermes.dsg.cs.tcd.ie/.

All code listings in this document are part of the Hermes Framework for the development of Mobile Context-aware Trails-based Applications. The Hermes Framework is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; version 3 of the License.

The Hermes Framework is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. A copy of the GNU Lesser General Public License is distributed with the Hermes Framework. It can be found in a file named COPYING.txt. If not, see <http://www.gnu.org/licenses/>.

Contents

1 PACKAGE CONTENTS 6

1.1 GISlite Library 7
1.1.1 Introduction 7
1.1.2 Features 8

1.2 Hermes Framework 9
1.2.1 Introduction 9
1.2.2 Architecture 9

1.3 Example Applications 11
1.3.1 Where Am I ? 11
1.3.2 Where Are We ? 11
1.3.3 RiddleHunt 12

2 DEVELOPMENT ENVIRONMENT 12

2.1 Hardware 12

2.2 Platform 12

2.3 IDE 13

2.4 Compiling 13
2.4.1 External Package dependencies 13
2.4.2 Internal Package Dependencies 14

3 BUILDING APPLICATIONS WITH HERMES 15

3.1 Configuring Hermes 15
3.1.1 Properties file 15
3.1.2 Initialising 16
3.1.3 Logging 16

3.2 Configuring GISlite 17
3.2.1 Properties file 17
3.2.2 Data Sources 19

3.3 Adding New Types of Context 19

3.4 Creating new Context Sources 21

3.5 Connecting Context Sources 22

3.6 Defining new types of Messages 22
3.6.1 Application-specific `CommunicationMessages` 23
3.6.2 Superclass for the application-specific `CommunicationMessages` 23
3.6.3 Subclass the CommunicationMessageFactory 23
3.6.4 Pass the subclass of the CommunicationMessageFactory into Hermes.init() 24

3.7 Using Application Specific Context Capsules 24
3.7.1 Creating Context Capsules 25
3.7.2 Add those ContextCapsules to the ContextCapsuleManager 26

3.7.3 Create any application-specific Content classes 26
3.7.4 Start the ContextCapsuleThread 27

3.8 Using Application specific Context Queries 27

3.9 Creating an interface for your application 28

3.10 Initialising your application 29
3.10.1 Call init() methods of Hermes and GISlite 29
3.10.2 Privacy settings 29
3.10.3 Setting up the Interface 29
3.10.4 Registering for updates in Context 30
3.10.5 Linking other ContextChangedEventListeners to the ContextManager 30
3.10.6 Adding ContextSources 31
3.10.7 Registering to receive ApplicationMessages 31
3.10.8 Creating new instance of the ContextCapsuleManager (if necessary) 31
3.10.9 Starting the ContextCapsuleThread (if necessary) 32

4 RELATED DOCUMENTS 33

5 APPENDIX 35

5.1 Code Listing 35

5.2 List of Figures 35

1 Package Contents

This section describes the contents of this package and the directory structures within the Hermes framework distribution.

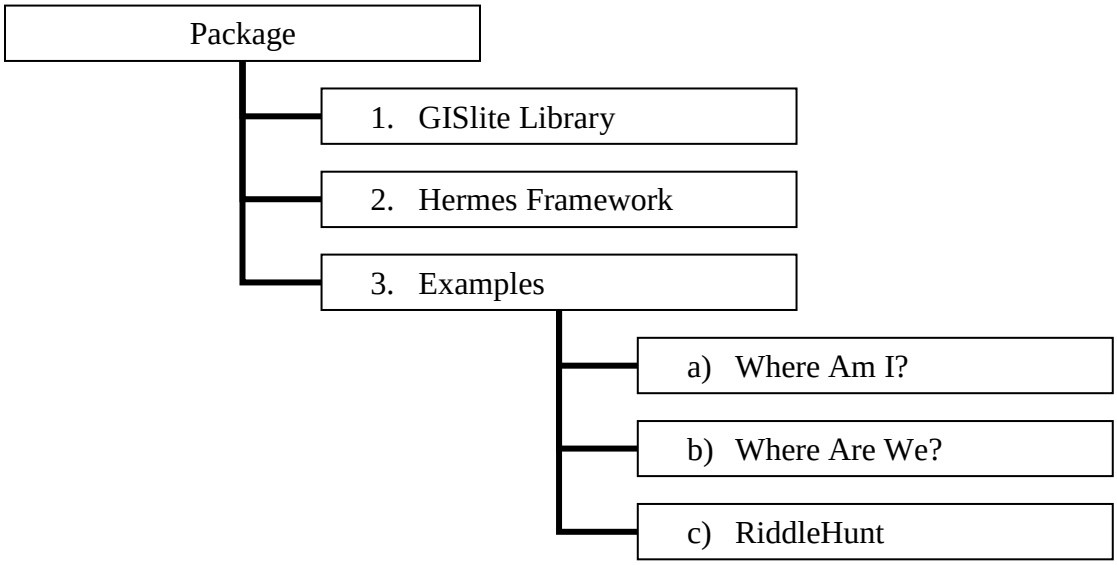

Figure 1 - Distribution Contents

1. The GISlite project is a standalone library for accessing, manipulating and displaying spatial data. This library is used by the example applications to provide an interface to display contextual information. This project contains the following sub directories;
    - `\bin` – The compiled java code
    - `\dist` – A packaged .jar or .jxe file containing all the compiled code and resources from the `\bin` directory
    - `\src` – the source code
    - `\doc` – Javadoc API documentation

2. The Hermes project contains the framework itself. The directory structure within the project is the same as for the GISlite project;
    - `\bin` – The compiled java code
    - `\dist` – A packaged jar / jxe file containing all the compiled code and resources from the `\bin` directory
    - `\src` – the source code
    - `\doc` – Javadoc API documentation

3. The Examples directory contains three example applications;
    - *Where Am I?* – Demonstrates the Hermes framework being used to gather 'location' context from a GPS context source and the GISlite project being used to display this information on a map.
    - *Where Are We?* – Extends the WhereAmI application to add a demonstration of the Hermes frameworks ability to share context between

devices. In this case the map display shows where neighbouring users are on the screen.

- *RiddleHunt* – A treasure hunt game involving the solving of riddles built using the Hermes framework.

Note: RiddleHunt code included here is incomplete. It is provided as an example of using the framework and is not a playable version of the game.

The following sections will introduce and describe the major features of each of the packaged libraries.

1.1 GISlite Library

GISlite is a helper library for Hermes that facilitates the generation of a map-based spatial context visualisation at different resolutions and levels of detail for mobile users.

1.1.1 Introduction

The GISlite library is designed to facilitate the rendering and reasoning over vector map data on mobile devices. The library is designed to be very simple to use and features an intuitive API. The API makes it easy to add additional layers to the map and to manipulate the map. For example map layers can be added to illustrate contextual information or custom controllers can be written to orientate and scale the map based on the user's location. GISlite also supports spatial reasoning and route generation. The following diagram illustrates the level at which the GISlite library is integrated into the component view of the Hermes framework.

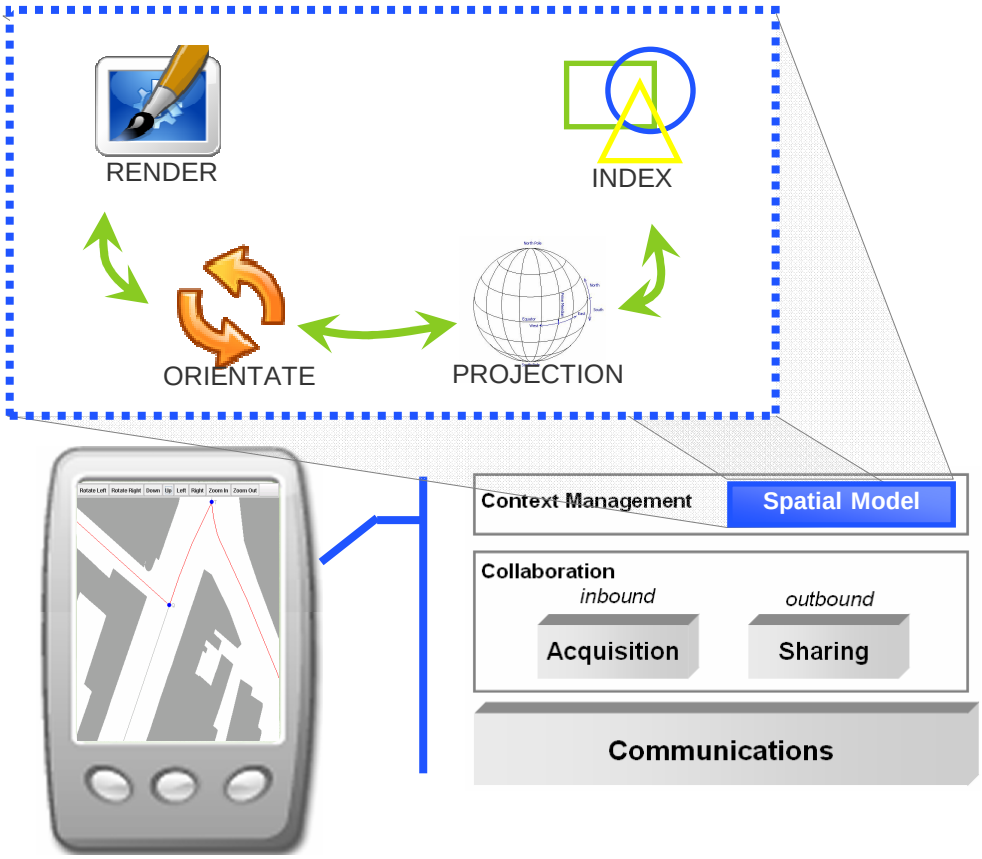

Figure 2 - How GISlite works with Hermes

1.1.2 Features

- Equirectangular and Cylindrical cartographic coordinate system projections.
- Vector data rendering and retrieval from ESRI Shapefiles[1].
- Multi-resolution spatial visualisation featuring scale sensitive level of detail and line simplification.
- Online continuous generalisation via nth algorithm for simple features (coast, lakes, rivers etc) and a variation of the Douglas-Peuker algorithm for complex features.
- Track-up orientated spatial visualisation.
- Rectangular polygon and line clipping to reduce the number of points to be drawn and further speed up rendering (Wriler-Atherton, Sutherland-Hodgeman and Liang-Barsky algorithms implemented).
- Shortest path topographical route generation.
- Quad-tree region based spatial indexing.
- Support's zoom, pan and rotate operations.
- Support for spatial operations such as containment, intersection, union, and distance on all features.

[1] ESRI Shapefiles store nontopological geometry and attribute information for spatial features. The geometry for a feature is stored as a shape comprising a set of vector coordinates. Shapefiles support point, line, and area features. Attributes are held in a dBASE® format file. Each attribute record has a one-to-one relationship with the associated shape record.

1.2 Hermes Framework

1.2.1 Introduction

Hermes is a software framework for mobile, context-aware trails-based applications which will support developers by providing generic components containing structure and behaviour common to all trails-based applications. At the most general level, a trail can be thought of as a collection of activities with associated locations, and a dynamically reconfigurable recommended visiting order. The trail is a collection of connected locations rather than a strict sequence since it may contain alternative sub-routes to cater for such variables as different modes of transport or user preferences. Trails underpin a wide range of useful applications for a mobile user who has a set of activities that may or should be carried out throughout the day at different locations. Combining the trails concept with mobile, context-aware technology creates opportunities for innovative activity-based application development. Mobile, context-aware applications are those that run on wireless devices e.g., PDAs and smartphones, and have an awareness of the physical and social situation in which they are deployed. Examples of trails applications that are both mobile and context-aware include tour guides, courier support/management systems, basic route planners, treasure hunt games and student support systems.

The following section will give an overview of the architecture of the framework.

1.2.2 Architecture

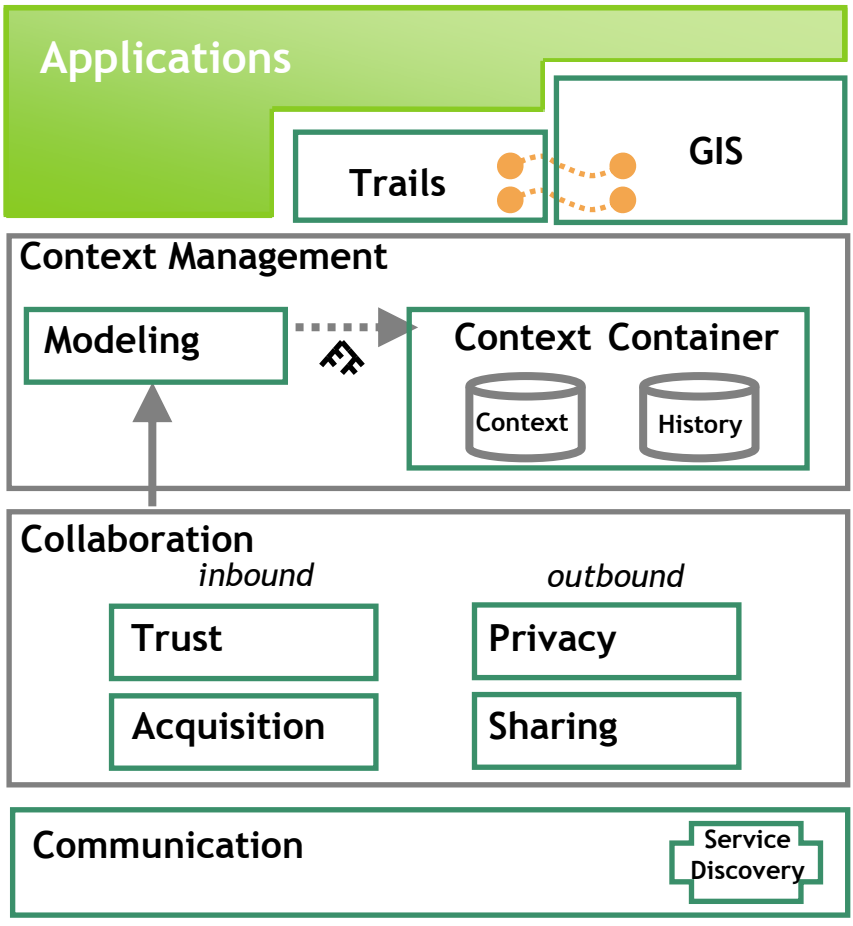

Figure 3 - Hermes framework component architecture.

Figure 3 - Hermes framework component architecture. shows a component level view of the of the Hermes framework. The diagram describes a layered software framework in which each layer is self contained and is isolated from the effects of changes in other layers through a combination of interfaces, event based call backs and message passing.

The architecture diagram should be read from a 'bottom-up' perspective as each ascending layer in the architecture builds upon the functionality of lower layers. At the bottom layer of the architecture is the *communication* component. It is the responsibility of this layer to manage the communication with sensors, infrastructure and peer devices over a variety of networks. This layer also manages bandwidth and the discovery of other devices supporting the Hermes framework.

Above communication are the components that support collaboration in the Hermes framework. They are seperated into columns to illustrate the separation between the inbound and outbound aspects of collaboration. On the inbound side (left) are the *acquisition* and *trust* components. The acquisition component is responsible for proactively seeking out and acquiring specific types of context on behalf of an application. The trust component works with acquisition to augment incoming messages with trust related metadata. On the outbound side of the collaboration components are the *sharing* and *privacy* components. The sharing component is responsible for responding to incoming requests for context and the privacy component provides for the protection of personal data by limiting which types of context are exchanged with which peers.

Above the collaboration layer is the context management layer containing the context *container* and *modelling* component. The modelling component handles the fusion of context and the conversion of incoming context to the context manager's internal representation. In addition to the modelling component there is a context container that is responsible for storing context persistently. The context container can hold context in memory or persist it to disk in response to the freshness of the context and the memory available on the device. The context container makes its context available by providing an interface for context queries and also making it possible for applications to register to be notified if a particular context changes.

It is at this level that many other context-aware toolkits end their support for application developers. In contrast Hermes contributes two additional closely collaborating components that provide solutions for a number of technical challenges common to trails-based applications. Theses components are the t*rails* component and *GIS* component. The trails component is responsible for the generation and management of trails. The GIS component is responsible for maintaining a model of the structure and geography of the user's environment for the purpose of supporting more intelligent reasoning by the trails component and improving the overall user experience of trails-based applications.

1.3 Example Applications

A number of example applications are included in the package. We list them here in order of complexity.

1.3.1 Where Am I ?

This is the first example application. It is very small (only two classes) and is intended to demonstrate the integration of GISlite and Hermes libraries by the addition of a map layer that overlays contextual information gathered by the Hermes framework over a GISlite map rendering.

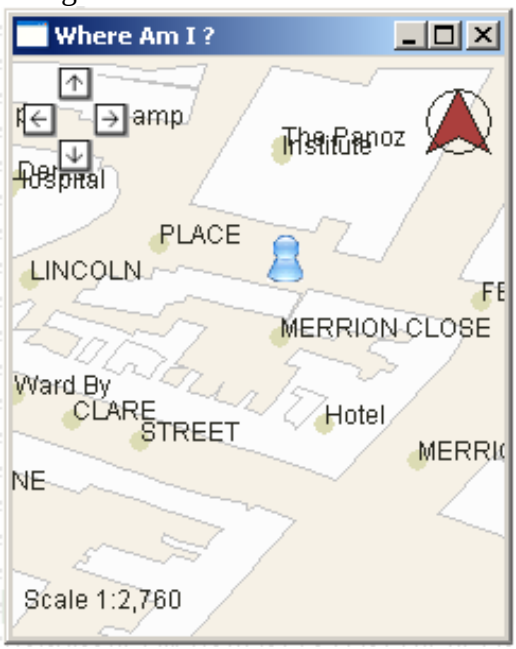

Figure 4 –*Where Am I ?* Interface

1.3.2 Where Are We ?

The second example builds on the first example. In this application in addition of producing a map layer showing a single piece of contextual information, a map layer showing the locations of all other people discovered by the Hermes framework on the map interface.

1.3.3 RiddleHunt

Figure 5 - Riddle Hunt

Riddle Hunt is a multiplayer, riddle solving game in which the application assists users in playing the game by both presenting context information such as other players' locations and by generating trails designed to help the player win the game. The version of the game included in this package is not the full version but is provided as a illustration of how to expand the capabilities of the framework components to handle extensions such as previously undefined types of context and new sensors.

2 Development Environment

This section provides the information necessary to build the source in this package.

2.1 Hardware

All the software in this package was designed to run on a PDA like device supporting a Java VM. The software has been successfully tested during development on HP IPAQ (h6300) devices running Windows Mobile 2003 and IBM Workplace Client Technology, Micro Edition JVM.

2.2 Platform

The Hermes framework is written in Java but is designed for the J9 Java VM. The J9 VM is an implementation of the Java Virtual Machine Specification, Version 1.3.

J9 VM can be downloaded as part of IBM Workplace Client Technology, Micro Edition 5.7 from the IBM website (requires registration). Included in this download is J9 (a.k.a IBM WebSphere Everyplace Micro Environment).

The following diagram illustrates the platform on which Hermes is designed to run.

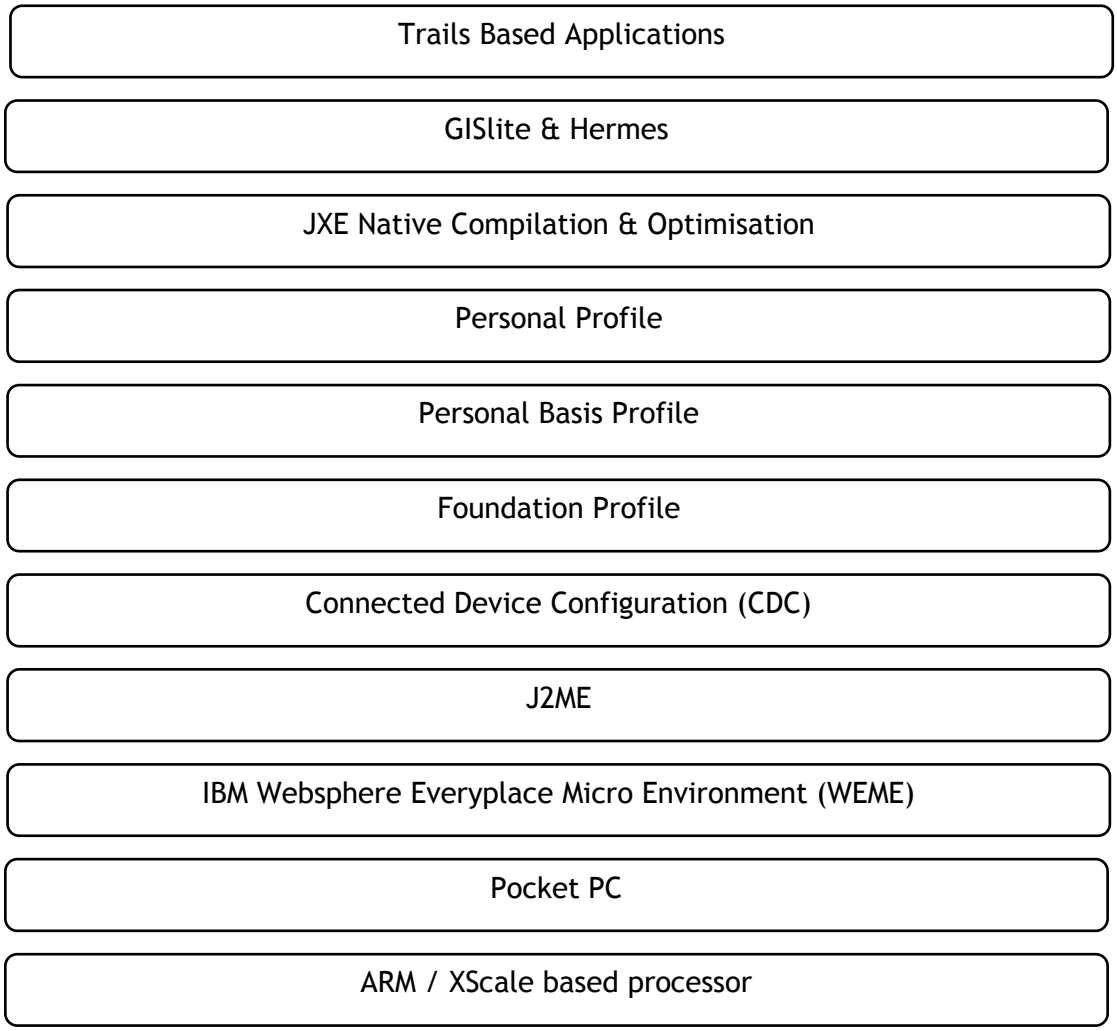

Figure 6 - Hermes Platform

2.3 IDE

All the code in this package was developed using WebSphere Studio Device Developer. Device Developer is an integrated development environment for the creation and testing of applications that will be deployed on handsets and other small devices.

2.4 Compiling

To compile all the source included in this distribution it is necessary to be aware of the compilation dependencies for each of the projects.

2.4.1 External Package dependencies

For each of the projects there are a number of external libraries which are required to compile and run the code. The table below lists these dependencies. In all instances the dependant libraries are included in the distribution and can be found in the /lib directory of the appropriate project.

Project	Jar	Detail
GISlite	n/a	n/a
Hermes	db4o-6.1-java1.1.jar kxml-min.jar RXTXcomm.jar	Object database engine. Very small memory footprint XmlPull Parser optimized for J2ME Provides access to serial ports in Java.

Table 1 – External Package dependencies

2.4.2 Internal Package Dependencies

The projects themselves are dependant on each other. In order to compile Hermes you will need to have GISlite on the build path. In order to compile any of the examples you will need both GISlite and Hermes on the build path.

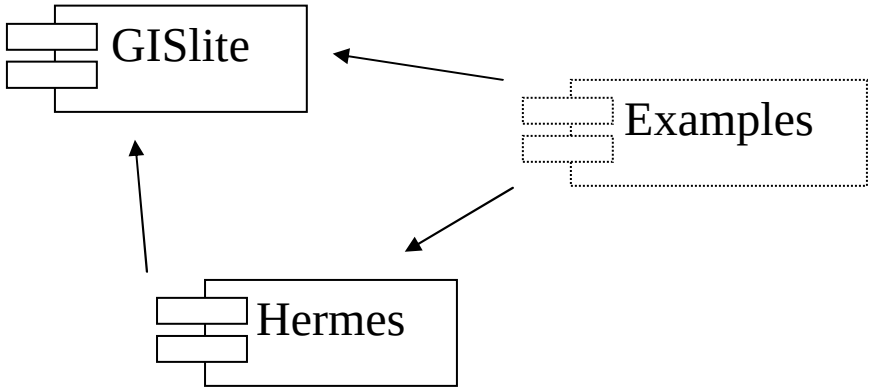

Figure 7 - Compilation dependencies

3 Building applications with Hermes

This section covers all the tasks necessary to build a context-aware trails-based application using a combination of GISlite and the Hermes framework.

3.1 Configuring Hermes

Many of the frameworks components are configurable in order to support different hardware platforms. This section covers everything needed to configure the framework for use by an application.

3.1.1 Properties file

A properties file is used to store all the configuration settings for the framework components. The framework requires this properties file to run and looks for it in the current working directory `/config/hermes.properties`. The properties file itself is a list of key value pairs. Each pair is on a separate line and lines beginning with # are comments.

```
# Hermes Properties File. DO NOT EDIT

# Communication properties
hermes.comms.threads.incoming=2
hermes.comms.samemachine=false
hermes.comms.heartbeattimeout=30000
hermes.comms.broadcast.port=4446
hermes.comms.broadcast.sleepinterval=10000
hermes.comms.listening.timeout=10000
hermes.comms.listening.sleep=5000
hermes.comms.receive.timeout=5000
hermes.comms.receive.sleep=5000
hermes.comms.incoming.sleep=2000

# The Serial Comms API to use (J2ME / PC / Linux)
hermes.comms.serial.provider=J5SerialConnection
#hermes.comms.serial.provider=J2MESerialConnection

# The Context Container to use
hermes.context.container.provider=DB4OContextContainer
#hermes.context.container.provider=InMemoryContextContainer

# Context Data Store
hermes.context.container.objectcontainer.file=data/db4o
#hermes.context.container.objectcontainer.file=Storage
Card/Hermes/data/db4o

# GPS (PC)
hermes.comms.gps.comm=4
hermes.comms.gps.baud=9600

# GPS (IPAQ)
# hermes.comms.gps.comm=7
# hermes.comms.gps.baud=9600

# Context Capsule properties
hermes.context.capsules.capsulesleeptime=5000
```

```
# Trail Generation
hermes.trail.subtrail.size=5
hermes.trail.reconfiguration.interval=10000

# the trail reconfiguration strategy
hermes.trail.reconfiguration.strategy=brute
#hermes.trail.reconfiguration.strategy=genetic
#hermes.trail.reconfiguration.strategy=annealing

hermes.trail.reconfiguration.relevence.threshold=0.8

# Debugging
hermes.debug.log.sysout=true
hermes.debug.log.file=data/hermes.log
#hermes.debug.log.file=Storage Card/Hermes/data/hermes.log
```

Listing 1 – Sample Hermes Properties File Contents

3.1.2 Initialising

The Hermes framework includes a static class that is used to initialise all the framework components, integrate these components where necessary and provide a single point of access to the framework for your application. To initialise the framework in your application the following statement should be executed.

```
Hermes.init(_appKey);
```

Listing 2 – Hermes Initialisation

The `_appKey` parameter uniquely identifies the application initializing the framework. This is used to distinguish among multiple applications accessing the framework in the case (while debugging) that multiple framework instances reside within the same JVM. For a specific application this string can be assigned to anything.

3.1.3 Logging

The Hermes framework includes a configurable logging utility that an application can call to print or record debugging information. To log information a statement similar to the following should be added.

```
Hermes.log.info("Hermes Framework (c) DSG 2007", Hermes.class);
```

Listing 3 - Logging

There are also `Hermes.log.debug` and `Hermes.log.error methods.` The methods take two parameters. The text to log and the class (if static) or object that the message is coming from. An example of how the Hermes logger may format this example would be.

```
INFO  01.08.2007 2:45:15 [Hermes] Hermes Framework (c) DSG
2007 (main)
```

The reason the class name is required is that the logging can be filtered by class. This is important because the volume of debugging information may be significant. For example it may be desirable to see all the debugging messages from just a single class. To allow for such configuration there is a properties file included as a resource in the build of Hermes that specifies which classes should export debugging information, which level of debugging is required (info, debug or error) and also whether to dump the logging information to the default output stream or write it to a file. An example of the default settings for this file is the following:

```
# RHLog Properties file. DO NOT EDIT

# Example:
#
# To Log from every class
# ie.tcd.cs.dsg.hermes.*=true
#
# To Log from every class except riddle hunt
# ie.tcd.cs.dsg.hermes.*=true
# ie.tcd.cs.dsg.hermes.app.RiddleHunt=true
#
# To log from only RiddleHunt class
# ie.tcd.cs.dsg.hermes.*=false
# ie.tcd.cs.dsg.hermes.app.RiddleHunt=true
#
# NOTE: Omitting a class name is equivalent to setting it to false

# PER CLASS LOGGING SETTINGS FOLLOW

ie.tcd.cs.dsg.hermes.*=true
ie.tcd.cs.dsg.hermes.comms.MessageInputStream=false

# END
```

Listing 4 – Default Logging Properties File

3.2 Configuring GISlite

This section covers the configuration of the GISlite library so that it may be used alongside Hermes to produce an interface to a trails-based application.

3.2.1 Properties file

GISlite requires its own properties file which must be placed in the path `/gislite/gislite.properties` from the current working directory. This file specifies the configuration options of the algorithms that are included in this library as well as specifying the data sources from which to retrieve spatial data. These data sources must be ESRI Shapefiles.

```
# GISlite Properties File.

# Debugging
gislite.debug=true

# Screen Size
gislite.screen.width=240
gislite.screen.height=300
```

```
# Initial Scale
gislite.scale=2760.3848
gislite.scale.cache.factor=2.5

# Initial Center of Map
gislite.center.lat=53.341442
gislite.center.lon=-6.2501383

# Data Sources
gislite.source.count=3
gislite.source.buffer=1024

gislite.source.3.type=shp
gislite.source.3.file=shapefiles\\country.shp

gislite.source.2.type=shp
gislite.source.2.file= shapefiles\\building_outlines.shp

gislite.source.1.type=shp
gislite.source.1.file=shapefiles\\text.shp

# Projection (EQUIRECTANGULAR)
gislite.projection=EquiRectangular
#gislite.projection=Orthographic
#gislite.projection=Lambert
#gislite.projection=Mercator

# Parametric Line & Polygon Clipping Algorithms
gislite.projection.clip.line=CohenSutherland
#gislite.projection.clip.line=LiangBarsky
gislite.projection.clip.poly=SutherlandHodgman
#gislite.projection.clip.poly=WeilerAtherton
#gislite.projection.clip.poly=Maillot
gislite.projection.clip.points.threshold=150
gislite.projection.clip.area.multiple=3

# Generalisation
gislite.geometry.filter.lod=MinimumAreaFilter
gislite.geometry.filter.lod.tolerance=20

# Geometry Simplification
gislite.geometry.generalise.tolerance=3
gislite.geometry.generalise.threshold=6000

# R-Tree Spatial Indexing
# > 40 * max node entries + 9
index.spatial.storage.pagesize=512
index.spatial.rtree.maxnodecapacity=12
index.spatial.rtree.minnodecapacity=5
index.spatial.rtree.nodesplit=Quadratic
#index.spatial.rtree.nodesplit=Linear
index.spatial.cache.size=8
```

Listing 5 – Sample GISlite Properties File

3.2.2 Data Sources

The data sources referred to in the above properties file must match the path to a '.shp' ESRI Shapefile. Examples of such files can be found in `gislite/shapefiles/`. A '.dbf' attribute record file is required for each shapefile but all other files in this directory will be generated if missing by GISlite. The following table lists the file extensions that may be found in this directory and the purpose of the files.

Note: As GISlite will generate files while running and place them in this folder, it must be writeable.

Extension	Description
SHP	Shapefile containing geometry. Multiple shapefiles may appear with similar names in this directory. These are multi-resolution caches of the original SHP file and can be safely removed at any time.
DBF	dBase Attribute database.
SHX	Shape Index – Flat index providing faster access to large SHP files.
SSX	Spatial Index – Flat index providing minimum bounding rectangle based access to SHP file records.
QIX	A Quad-tree spatial index.

Table 2 - Data source file extensions.

3.3 Adding New Types of Context

To demonstrate adding new types of Context to an application, we will use examples from RiddleHunt's application-specific Contexts: Game and Player. Game extends the abstract Context, Artefact and Player extends the concrete Context, Person. Several methods need to be overridden.

- A protected constructor without arguments. The constructor is protected because all Context should be created with the following ContextFactory method:

```
public Context createNewContext(String type)
```

Listing 6 – The signature of the ContextFactory method for creating Context

- All outside access to a Context object should be done through setter methods. This is particularly important, as the Context objects must keep track of when they were last updated in order to aid in recognizing stale Context, among other considerations. This timestamp maintenance is carried out by the inclusion of an `updateTimestamp()` call in all the setters. An example from Player follows:

```
public void setScore(int i) {
            updateTimestamp();
            _score = i;
}
```

Listing 7 – An example of a setter that calls updateTimestamp()

As a consequence of having the `updateTimestamp()` call in the setters, it must be disabled when using the setters and the timestamp should not be updated. This is done by setting the Context attribute _shouldSetTimestamp to false at the

beginning of those methods and setting it back to true after all the setters have been called. This should be done for any methods that want to set the attributes of the Context but do not wish to update the timestamp, e.g., the clone, merge, and fromXML methods all with to populate a Context object's attributes but do not wish to update the timestamp. For an example see the clone and merge discussion below.

- A clone method must be implemented if a deep copy of any structure is required. The clone() method from the Trail Context is shown below. A deep copy of the Trail's Activity array and Activity HashMap is required. Notice the _shouldSetTimestamp statements described above surrounding the setter:

```
public Object clone() {
	Trail clonedTrail = (Trail) super.clone();
	clonedTrail._shouldSetTimestamp = false;

	// A deep copy of the Activity array and the Activity map
	Activity[] clonedActivities = null;
	if (activities != null)
	{
		clonedActivities = (Activity[]) activities.clone();
	}
	clonedTrail.setActivities(clonedActivities);

	clonedTrail._shouldSetTimestamp = true;
	return clonedTrail;
}
```

Listing 8 – An example of the clone() method for the Trail context

- `toXML()` and `fromXML()` for converting Context to and from XML for use in CommunicationMessages. As mentioned above, `fromXML()` needs turn off any timestamp update if it uses a setter.
- `merge()` also needs to dictate what behavior to execute when merging two Contexts of the same type. E.g., a common merging strategy is to populate the Context with the values of whatever Context has the most recent timestamp. As mentioned above, `merge()` should turn off any timestamp update if it uses a setter.

```
public boolean merge(Context c) {
	boolean changed = false;

	if (c instanceof Player) {
		Player player = (Player) c;

		_shouldSetTimestamp = false;
		// If the above was merged that means,
		    // there should be some new data
		if (super.merge(player))
		{
			this.setScore(player.getScore());
			this.setStrategy(player.getStrategy());

			this.setIsGamesMaster(player.isGamesMaster());
			changed = true;
		}
```

```
            _shouldSetTimestamp = true;
      }

      return changed;
}
```

Listing 9 – An example merge() method for subclasses of Context

Adding new types of Context requires an extension of a few classes beyond the actual Context classes.

- `ContextType` should be extended to store the static Strings representing the new context types. In RiddleHunt, the class extending `ContextType` is `RHContextType` and contains the Strings:

```
public static final String GAME = "Game";
public static final String PLAYER = "Player";
```

Listing 10 – Context type String attributes of RHContextType

- In addition, `ContextFactory` should be extended to add the new Context types to the list of Context currently available and allows the new types to be created via the `ContextFactory`. This is done by adding the new types to the `ContextFactory` data structures via the `addNewContextType()` call. The first argument is the Context type's String, the second is a prototype for creating new Context objects, and the last designates this context type as abstract or not. In RiddleHunt, the `RHContextFactory` has these calls in its constructor to add the `Game` and `Player` types:

```
// Add any types that are not added by the super classes
addNewContextType(RHContextType.GAME, new Game(), false);
addNewContextType(RHContextType.PLAYER, new Player(), false);
```

Listing 11 – Calls made in RHContextFactory constructor to add new types of Context

- Finally, if any of the new types require a special `ContextQuery` to retrieve context, this needs to be created by extending the `ContextQueryFactory`, as described in Section 3.8.

3.4 Creating new Context Sources

The framework is designed so that new context types and context sources can be easily added. It is a simple process to implement the code to support a new type of context source. Context sources can be anything from a new type of sensor to an online service.

Hermes contains a GPS context source ready to use and this code serves as an example of how to create your own context sources:

```
public class GPSReceiver extends ContextSource {

      /** The ContextTypes that GPS receivers can deliver */
      private static final String [] _gpsContextTypes =
```

```
                                        { ContextType.LOCATION};

      private static final ContextServiceDescription[]
_contextServiceDescriptions =
 ContextServiceDescription.getDescriptionsFromType(_gpsContextTypes,
true);

      public GPSReceiver(SerialAddress serialPortAddress) {
            // Setting the last time this was updated to
            // be MIN_VALUE, so it will always get updated
            super(new Device("GPS Receiver", serialPortAddress),
     _contextServiceDescriptions, System.currentTimeMillis(), 5000);
      }
}
```

Listing 12 – Creating a ContextSource

The important aspects of this code listing is that every context source must have a reference to an underlying instance of `Device`, a set of service descriptions and a period to control the polling of the device. The service descriptions define the types of context available from this device (in this case just location) and their format. This information is used by the framework to determine how to format and model the information as context.

3.5 Connecting Context Sources

Once the decision of which context sources should be available to the application is made and the code extending the generic `ContextSource` class for each one is written, it is necessary to inform the framework that at runtime it should connect to the context source and acquire its context. This is facilitated by a single call to the static `Hermes` class.

```
SerialAddress serialPort = new SerialAddress(
                                        (short) commPort, baudRate);
GPSReceiver gps = new GPSReceiver(serialPort);

ContextServiceDescription[] sd =
  (ContextServiceDescription[]) gps.getContextServiceDescriptions();

// Discover the context Source
Hermes.getCommunication().getServiceDiscovery().updateContextSourceLi
st(gps);
```

Listing 13 – Adding a Context Source to the application

3.6 Defining new types of Messages

If new types of `CommunicationMessage` are needed, four steps need to be carried out:

1. Create the application-specific `CommunicationMessage`
2. Create a superclass for the application-specific `CommunicationMessage`
3. Create a subclass of the `CommunicationMessageFactory`
4. Pass the subclass of the `CommunicationMessageFactory` into `Hermes.init()`.

3.6.1 Application-specific `CommunicationMessages`

A subclass of `CommunicationMessage` needs to be created for every new concrete type of message an application requires. These need to override the `messageHelper()` method that converts its data to XML and contain a `createXXXMessage()` method that creates a new message from XML, where "XXX" stands for the specific type of message. In RiddleHunt, the only new message is the `StartGameApplicationMessage` that extends `RHApplicationMessage` (see below). This class overrides `messageHelper()` and contains the method `createStartGameApplicationMessage()`.

3.6.2 Superclass for the application-specific `CommunicationMessages`

- If the new `CommunicationMessage` is a subclass of `ServiceMessage`, `ContextMessage`, or `ApplicationMessage`, that superclass itself must be extended. These new subclasses direct the messages of the newly created types to their corresponding classes with their corresponding `createXXXMessage()` methods. In RiddleHunt, the `ApplicationMessage` is extended by `RHApplicationMessage`, which directs any `StartGameApplicationMessages` to the correct method:

```
public static RHApplicationMessage
     createApplicationMessage(XMLElement element) {
      RHApplicationMessage retMessage = null;

      String messageTypeString =
                           element.getAttributeValue("messageType");
      if (messageTypeString != null) {
            int messageType;
            try {
                  messageType = new
                              Integer(messageTypeString).intValue();
            } catch (NumberFormatException nfe) {
                  messageType = UNKNOWN_MESSAGE_TYPE;
            }

            if (messageType == START_GAME_TYPE) {
                  retMessage = StartGameApplicationMessage
                        .createStartGameApplicationMessage(element);
            } else {
                  Hermes.log.error(
                  "createApplicationMessage: Unknown Message Type = "
                  + messageType,    RHApplicationMessage.class);
            }
      }
      return retMessage;
}
```

Listing 14 – Directing the input XML to the correct RHApplicationCommunicationMessage class

3.6.3 Subclass the CommunicationMessageFactory

The application must provide a method for the framework to create an application-specific method if it receives one. To do this, the

`CommunicationMessageFactory` must be extended and the `createSpecificMessage()` method must be overridden. This method handles the creation of any of the application-specific messages by calling the correct `createXXXMessage()` method. This will be a static method of a specific concrete `CommunicationMessage` or an extension of an existing abstract superclass (e.g., in RiddleHunt, `RHApplicationMessage` extends the abstract `ApplicationMessage`). If the type of message passed in is not a newly created type, the input is passed on to the `createSpecificMessage()` method of the superclass. In RiddleHunt, `RHCommunicationMessageFactory` extends `CommunicationMessageFactory` which contains the overridden `createSpecificMessage()` method. In this case, the only application-specific `CommunicationMessage` is an `RHApplicationMessage`. The other `CommunicationMessages` are passed on to the superclass (`CommunicationMessageFactory`):

```
public CommunicationMessage createSpecificMessage(
                          int messageClass, XMLElement element) {

      CommunicationMessage retMessage = null;

      if (messageClass ==
                CommunicationMessage.APPLICATION_MESSAGE_CLASS) {

         retMessage =
            RHApplicationMessage.createApplicationMessage(element);
      } else {
            retMessage = super.createSpecificMessage(
                                              messageClass, element);
      }
      return retMessage;
}
```

Listing 15 – Subclass of CommunicationMessageFactory

3.6.4 Pass the subclass of the CommunicationMessageFactory into Hermes.init()

Finally, a new instance of the subclass of `CommunicationMessageFactory` must then be passed as a parameter to the `init()` method of `Hermes` in order to assign it as the `CommunicationMessageFactory` for the application.

3.7 Using Application Specific Context Capsules

Much like a time capsule gives a clearer picture of the time period at which it was created, a context capsule stores content, along with Surrounding Context Information (SCI) that allows the application to interpret that content in its original context. This content can be anything an application can interpret, such as a behavior or a piece of context. Context capsules allow an application to activate its content when its SCI is sufficiently similar to the device's current context, e.g., if the context capsule's location in its SCI is similar to the device's current location. For more information on

context capsules please refer to the paper "Preserving Context with Context Capsules" listed in the Related Documents section of this document.

In the Hermes framework, the `ContextManager` periodically compares the device's current context to the SCI of each `ContextCapsule`. If the contexts are similar, the `ContextCapsule`'s `Content` is activated. The `Content` can be a piece of context such as an `Activity` or a `Trail` (`ContextContent.class`) or a behavior such as turning on a light or presenting a collection of riddles (`BehaviorContent.class`).

To use context capsules in a new application, four tasks need to be carried out:

1. Create any `ContextCapsules` necessary for the application.
2. Add those `ContextCapsules` to the `ContextCapsuleManager`'s `ContextCapsule` collection
3. Create any application-specific `Content` classes.
4. Start the `ContextCapsuleThread`

3.7.1 Creating Context Capsules

The RiddleHunt code below creates ContextCapsules that let the application know that riddles associated with a `Location` are available to be attempted when the player's location intersects the capsule's location. The activated content is an application-specific `BehaviorContent`, `RHBehaviorContent` with behavior type `RHBehaviorContent.RIDDLE_LOCATION_ACTIVATED_TYPE`, that is described later in this section.

```
// Set up geometry that if a Location is within the Geometry
// the capsule can be activated
Location bBox = (Location)
Hermes.getContextFactory().createNewContext(RHContextType.LOCATION);
bBox.setLocationGeometry(
      new BoundingBox(location.getLocationAsPoint(),
RIDDLE_SQUARE_SIDE_LENGTH, RIDDLE_SQUARE_SIDE_LENGTH));

// Create the Content which is a RiddleLocationActivated
BehaviorContent bContent =
      new BehaviorContent(new Object[] { location },
RHBehaviorContent.RIDDLE_LOCATION_ACTIVATED_TYPE);

// Create and add the capsule to the CapsuleStaging and location to
context capsule map
ContextCapsule capsule =
      new ContextCapsule(
            new Content[] { bContent },
            new Context[] { bBox },
            Communication.getLocalAddress(),
            TIME_UNTIL_REACTIVATION);
_locationToContextCapsules.put(location, capsule);
addContextCapsule(capsule);
```

Listing 16 – Creation of ContextCapsules for the Riddles (From RiddleManager's constructor)

As another example of a context capsule, the following code from RiddleHunt creates an "end of game" capsule that notifies the application that the game should end. The capsule is available to be activated when the current time is equal to or later than the time in the capsule. The activated content is an application-specific `BehaviorContent`, `RHBehaviorContent` with behavior type `RHBehaviorContent.END_GAME_TYPE`, that is described below.

```
// Now add the EndGame capsule
_gameDurationInMinutes = Hermes.getProperties().getPropertyAsInt(
                         "hermes.riddlehunt.gamedurationinminutes");

_timeGameOver = _gameDurationInMinutes * 60 * 1000 +
System.currentTimeMillis();

BehaviorContent endGameContent =
                  new BehaviorContent(new Object[] {},
                                      RHBehaviorContent.END_GAME_TYPE);
ContextCapsule endGameCapsule =
              new ContextCapsule(new Content[] { endGameContent },
                   new Context[] {},
                              Communication.getLocalAddress(), -1);

endGameCapsule.setTimeUntilReactivation(_timeGameOver);
addContextCapsule(endGameCapsule);
```

Listing 17 – Creation of the "end of game" ContextCapsule

3.7.2 Add those ContextCapsules to the ContextCapsuleManager

Add those ContextCapsules to the ContextCapsuleManager's context capsule collection with the `addContextCapsule(ContextCapsule capsule)`. E.g., the "end of game" `ContextCapsule` created above is added with the following line:

```
addContextCapsule(endGameCapsule);
```

Listing 18 – Addition of "end of game" ContextCapsule

3.7.3 Create any application-specific Content classes

If an application uses ContextCapsules, new application-specific Content will probably need to be created. RiddleHunt uses a new `BehaviorContent` called `RHBehavior` that defines the types for the context capsules described in the previous sections.

```
/** Riddles are available to be presented */
public static final int RIDDLE_LOCATION_ACTIVATED_TYPE = 1;
/** The game is over */
public static final int END_GAME_TYPE = 2;
```

Listing 19 – The behavior types for RHContentBehavior

Once the content of a `ContextCapsule` has been activated, the Content is sent to the `ContextCapsuleHandler`. The RiddleHunt class is a

`ContextCapsuleHandler` and handles the previously described Content as follows:

```
public void handleBehaviorContent(BehaviorContent behavior,
                    Context[] sci, DeviceAddress creatorsDevice) {
      if (behavior.getBehaviorType() ==
                RHBehaviorContent.RIDDLE_LOCATION_ACTIVATED_TYPE) {
            // The first param is a Location (Location) of the Riddle
            Object obj = behavior.getParam(0);
            if (obj != null && obj instanceof Location) {
                  Location riddleLocation = (Location) obj;
                  _riddleManager.presentRiddles(riddleLocation);
            } else {
                  Hermes.log.error("First Behavior parameter should
                                        be a Spatial Feature", this);
            }
      } else {
            Hermes.log.debug("Unknown behavior type", this);
      }
}
```

Listing 20 – RiddleHunt's method to handle ContextCapusle BehaviorContent

3.7.4 Start the ContextCapsuleThread

Start the `ContextCapsuleThread` with ContextCapsuleManager's `launchContextCapsuleThread()`:

```
// Now launch the thread
super.launchContextCapsuleThread();
```

Listing 21 – Running the ContextCapsuleThread

3.8 Using Application specific Context Queries

The `ContextQueryFactory` allows new context queries to be created. If a new context type requires a `ContextQuery` different than the standard `ContextTypeQuery` in the Hermes framework, a new ContextQuery should be created. The `ContextQueryFactory` should also be extended. In RiddleHunt, `RHContextQueryFactory` does this by overriding the `ContextQueryFactory`'s `createContextTypeQuery()` method to call the newly created PlayerQuery's constructor when encountering a Player type:

```
public ContextQuery createContextTypeQuery(String contextType,
                                                String name) {
      ContextQuery returnQuery = null;
      if (contextType.equals(RHContextType.PLAYER)) {
            returnQuery = new PlayerQuery(name);
      } else {
            returnQuery = super.createContextTypeQuery(
                                          contextType, name);
      }
```

```
      return returnQuery;
}
```

Listing 22 – Creating an application specific context query.

3.9 Creating an interface for your application

The GISlite library provides a simple map based interface that can be easily extended to produce an interface for your application. This is achieved by adding layers to the map display. This is supported by the `MapLayer`. An example of a very simple `MapLayer` follows. Once created, your custom map layers can be added to the map interface by calling `GISlite.getMapContext().addLayer()`.

```
public class ContextMapLayer extends DynamicMapLayer implements
ContextChangedEventListener {

      /** Cache of the users location so that UI can be rendered
       without querying context model every time */
      private Point dotLocation;

      public ContextMapLayer(RotatableProjection projection,
                              Style style) {
            super("Context Layer", projection, style);

            // Try to get curent location
            ContextQuery q =
               Hermes.createContextTypeQuery(ContextType.LOCATION,
                 Hermes.MY_NAME);
            Context[] c = Hermes.getContext(q);
            if (c != null && c.length > 0)
                  dotLocation = ((Location)
                        c[0]).getLocationAsPoint();
      }

      public void render(Graphics g) {
            if (dotLocation != null) {
                  dotLocation.render(g, this._projection,
                      StyleImpl.HIGHLIGHT_STYLE);
            }

      }

      public void handleContextChangedEvent(
                                 ContextChangedEvent event) {

            if (event.getContext().getContextType().equals(
                                       ContextType.LOCATION)) {

                  // Update 'myLocation' Point
                  Location loc = (Location) event.getContext();
                  this.dotLocation = loc.getLocationAsPoint();

                  // Trigger Redraw layer and center if necessary
                  this.fireLayerChangedEvent();
            }
      }
}
```

Listing 23 – Extending Map Layer in your application

For more information on `MapLayers` please consult the API documentation that accompanies GISlite.

3.10 Initialising your application

The following steps should be done to create a Hermes application. All of the sample code comes from `RiddleHunt.class`.

3.10.1 Call init() methods of Hermes and GISlite

The application code initializes Hermes and GISlite through their static `init()` calls. If you've extended any of the factories mentioned above (`ContextFactory`, `CommunicationMessageFactory`, or `ContextQueryFactory`), new instantiations of these classes should be passed into the `init()` method of Hermes. This constructor for RiddleHunt begins by calling both `init()` methods with the appropriate parameters:

```
GISlite.init(); // In Hermes properties file too

_selfName = GISlite.getProperties().getProperty(PERSON_NAME_KEY);

// Need to pass in all the hooks for the application
Hermes.init(_selfName, new RHContextFactory(), new
RHCommunicationMessageFactory(), new RHContextQueryFactory());

// User the Hermes Logger for debugging messages
GISlite.log = Hermes.log;
```

Listing 24 – Calling init() methods of Hermes and GISlite

3.10.2 Privacy settings

After calling the `init()` methods, the privacy settings for each type should be set using `Privacy`'s `addContextToDisclose()` method. This currently only allows disclosure of context based on type. The application should call this method for every type of `Context` that it wishes to share with other users. Any type not explicitly disclosed, will not be shared by the application. RiddleHunt shares the `Game`, `Player`, and `Location` contexts with other devices in the game:

```
// Configure what Context types we are willing to show and how often
Hermes.getPrivacy().addContextTypeToDisclose(ContextType.LOCATION,
1000);
Hermes.getPrivacy().addContextTypeToDisclose(RHContextType.PLAYER,
1000);
Hermes.getPrivacy().addContextTypeToDisclose(RHContextType.GAME,
1000);
```

Listing 25 – Set all the privacy settings

3.10.3 Setting up the Interface

An interface (created using the GISlite Library API) is then added to the application. The code snippet below also shows a map centring control being used to keep the map centred on the data in the RiddleHunt layer.

```
// Add ContextLayer to Map
RiddleHuntMapLayer contextLayer =
      new RiddleHuntMapLayer(GISlite.getMapContext().getProjection(),
                             StyleImpl.GRID_LAYER_STYLE);

// Hook up Map Centering Code
centerMap = new MapCentering(GISlite.getMapContext(), contextLayer);

// Have map centering listen for moving points
contextLayer.addLayerListener(centerMap);

GISlite.getMapContext().addLayer(contextLayer);
```

Listing 26 – Configuring the ContextLayer

3.10.4 Registering for updates in Context

If the application is interested in receiving any `ContextChangedEvents`, it should implement the `ContextChangedEventListener` interface. The application should then register its interest in any context types by calling the `Hermes.setInterestInContextTypes()` method. For RiddleHunt, the application is interested in receiving new `Player` and `Game` contexts:

```
// Subscribe to changes in Game Context and Instruct Acquisition to
acquire Game Context
Hermes.setInterestInContextTypes(this, RHContextType.PLAYER, 100);
Hermes.setInterestInContextTypes(this, RHContextType.GAME, 100);
```

Listing 27 – Registering for the types of Context needed by the application

3.10.5 Linking other ContextChangedEventListeners to the ContextManager

The application should also link any other `ContextChangedEventListeners` that are interested in receiving `Context` updates. In RiddleHunt, the `RiddleHuntMapLayer` is interested in receiving `Trail` updates.

```
// So the contextLayer will show activies on the map
Hermes.setInterestInContextTypes(contextLayer, RHContextType.TRAIL,
100);
```

Listing 28 – Linking the ContextLayer containing the Activities to the ContextManager

3.10.6 Adding ContextSources

Any local `ContextSources` such as GPS devices should be initialized. In RiddleHunt, the Location is generated from an attached GPS device. This `ContextSource` is added through a call to ServiceDiscovery's `updateContextSourceList()` method:

```
SerialAddress serialPort = new SerialAddress(
                                        (short) commPort, baudRate);
GPSReceiver gps = new GPSReceiver(serialPort);

gpsTypes = (ContextServiceDescription[])
                        gps.getContextServiceDescriptions();

// Discover the context Source
Hermes.getCommunication().getServiceDiscovery()
                                    .updateContextSourceList(gps);
```

Listing 29 – Adding a GPS ContextSource

3.10.7 Registering to receive ApplicationMessages

The application must also implement the `ApplicationMessageEventListener` interface and register with Communication if it wants to receive any application messages. In RiddleHunt, the application extends this interface and registers through the following call:

```
// We want to listen for application messages
Hermes.getCommunication().addApplicationMessageEventListener(this);
```

Listing 30 – Register to receive ApplicationMessages

3.10.8 Creating new instance of the ContextCapsuleManager (if necessary)

Initialize and start ContextCapsule thread (if necessary). In RiddleHunt, the ContextCapsuleManager is the RiddleManager. The RiddleManager is created in the application constructor and the ContextCapsuleHandler is passed as an argument (in this case, RiddleHunt implements the ContextCapsuleHandler interface).

```
// Starting Riddle Manager
_riddleManager = new RiddleManager(activities, this);
```

Listing 31 – Create a new instance of the ContextCapsuleManager

3.10.9 Starting the ContextCapsuleThread (if necessary)

The ContextCapsuleThread is started once the game has begun:

```
// Starts thread that loops through all ContextCapsules
_riddleManager.launchContextCapsuleThread();
```

Listing 32 – Run the ContextCapsuleThread

4 Related Documents

- Cormac Driver. "An Application Framework for Mobile, Context-Aware Trails. ", PhD thesis, Trinity College Dublin, 2007.
- Siobhán Clarke, Mike Spence, Cormac Driver and Éamonn Linehan "Using Collaborative Context for Dynamic Mobile Trails Generation", Intel Research Council Year 3 Renewal Document, May 2006.
- Mike Spence, Siobhán Clarke. "Preserving Context with Context Capsules." The Third International Workshop on Modeling and Retrieval of Context, AAAI 2006, Boston, MA.
- Cormac Driver, Éamonn Linehan and Siobhán Clarke, "Analysis of the Evaluation of Application-Led Research in Pervasive Computing", Technical Report, TCD-CS-2006-26, 4 May 2006.
- Mike Spence, Cormac Driver, Siobhán Clarke. "Sharing Context History in Mobile, Context-Aware Trails-Based Applications" 1st International Workshop on Exploiting Context Histories in Smart Environments, (ECHISE 2005) Pervasive 2005, Munich, Germany.
- Éamonn Linehan, Cormac Driver, Siobhán Clarke. "Route Generation for Adaptable Trails-Based Applications". 3rd Uk-UbiNet Workshop: "Designing, Evaluating and using Ubiquitous Computing Systems". University of Bath, February 2005.
- Mike Spence, Cormac Driver, Siobhán Clarke. "Collaborative Context in Mobile, Context-Aware Trails-Based Applications". 3rd Uk-UbiNet Workshop: "Designing, Evaluating and using Ubiquitous Computing Systems". University of Bath, February 2005.
- Cormac Driver, Éamonn Linehan, Siobhán Clarke, Andrew Jackson, Shiu Lun Tsang, Mike Spence: "A Framework for Mobile, Context-Aware Trails-based Applications: Experiences with an Application-led Approach". Submitted to the "What makes for good application-led research in ubiquitous computing?“ Workshop at Pervasive 2005, Munich, Germany.
- Siobhán Clarke, Cormac Driver. "Context-Aware Trails". IEEE Computer, Vol. 37, No. 8. pp. 97-99, August 2004. Invisible Computing column.

- Cormac Driver, Siobhán Clarke. "*Hermes*: Generic Designs for Mobile, Context-Aware Trails-based Applications". Workshop on Context Awareness at MobiSys 2004, Boston.
- Cormac Driver, Siobhán Clarke. "*Hermes*: A Software Framework for Mobile, Context-Aware Trails". Workshop on Computer Support for Human Tasks and Activities at Pervasive 2004, Vienna.

5 Appendix

5.1 Code Listing

LISTING 1 – SAMPLE HERMES PROPERTIES FILE CONTENTS 16
LISTING 2 – HERMES INITIALISATION 16
LISTING 3 - LOGGING 16
LISTING 4 – DEFAULT LOGGING PROPERTIES FILE 17
LISTING 5 – SAMPLE GISLITE PROPERTIES FILE 18
LISTING 6 – THE SIGNATURE OF THE CONTEXTFACTORY METHOD FOR CREATING CONTEXT 19
LISTING 7 – AN EXAMPLE OF A SETTER THAT CALLS UPDATETIMESTAMP() 19
LISTING 8 – AN EXAMPLE OF THE CLONE() METHOD FOR THE TRAIL CONTEXT 20
LISTING 9 – AN EXAMPLE MERGE() METHOD FOR SUBCLASSES OF CONTEXT 21
LISTING 10 – CONTEXT TYPE STRING ATTRIBUTES OF RHCONTEXTTYPE 21
LISTING 11 – CALLS MADE IN RHCONTEXTFACTORY CONSTRUCTOR TO ADD NEW TYPES OF CONTEXT 21
LISTING 12 – CREATING A CONTEXTSOURCE 22
LISTING 13 – ADDING A CONTEXT SOURCE TO THE APPLICATION 22
LISTING 14 – DIRECTING THE INPUT XML TO THE CORRECT RHAPPLICATIONCOMMUNICATIONMESSAGE CLASS 23
LISTING 15 – SUBCLASS OF COMMUNICATIONMESSAGEFACTORY 24
LISTING 16 – CREATION OF CONTEXTCAPSULES FOR THE RIDDLES (FROM RIDDLEMANAGER'S CONSTRUCTOR) 25
LISTING 17 – CREATION OF THE "END OF GAME" CONTEXTCAPSULE 26
LISTING 18 – ADDITION OF "END OF GAME" CONTEXTCAPSULE 26
LISTING 19 – THE BEHAVIOR TYPES FOR RHCONTENTBEHAVIOR 26
LISTING 20 – RIDDLEHUNT'S METHOD TO HANDLE CONTEXTCAPUSLE BEHAVIORCONTENT 27
LISTING 21 – RUNNING THE CONTEXTCAPSULETHREAD 27
LISTING 22 – CREATING AN APPLICATION SPECIFIC CONTEXT QUERY. 28
LISTING 23 – EXTENDING MAP LAYER IN YOUR APPLICATION 28
LISTING 24 – CALLING INIT() METHODS OF HERMES AND GISLITE 29
LISTING 25 – SET ALL THE PRIVACY SETTINGS 29
LISTING 26 – CONFIGURING THE CONTEXTLAYER 30
LISTING 27 – REGISTERING FOR THE TYPES OF CONTEXT NEEDED BY THE APPLICATION 30
LISTING 28 – LINKING THE CONTEXTLAYER CONTAINING THE ACTIVITIES TO THE CONTEXTMANAGER 30
LISTING 29 – ADDING A GPS CONTEXTSOURCE 31
LISTING 30 – REGISTER TO RECEIVE APPLICATIONMESSAGES 31
LISTING 31 – CREATE A NEW INSTANCE OF THE CONTEXTCAPSULEMANAGER 31
LISTING 32 – RUN THE CONTEXTCAPSULETHREAD 32

5.2 List of Figures

FIGURE 1 - DISTRIBUTION CONTENTS 6
FIGURE 2 - HOW GISLITE WORKS WITH HERMES 8
FIGURE 3 - HERMES FRAMEWORK COMPONENT ARCHITECTURE. 9
FIGURE 4 –*WHERE AM I ?* INTERFACE 11
FIGURE 5 - RIDDLE HUNT 12
FIGURE 6 - HERMES PLATFORM 13
FIGURE 7 - COMPILATION DEPENDENCIES 14